The subscripts in this example are numeric, but one of the most useful features of Awk is that array subscripts are not limited to numeric values; they can be arbitrary strings of characters. We will illustrate such subscripts in later chapters.

1.8 Useful One-liners

Although Awk can be used to write programs of some complexity, many useful programs are no more complicated than what we've seen so far. Here is a collection of short programs that you might find handy and/or instructive. Most of them are variations on material we have already covered.

Print the total number of input lines:

```
END { print NR }
```

Print the first 10 input lines:

```
NR <= 10
```

Print the tenth input line:

```
NR == 10
```

Print every tenth input line, starting with line 1:

```
NR % 10 == 1
```

Print the last field of every input line:

```
{ print $NF }
```

Print the last field of the last input line:

```
END { print $NF }
```

Print every input line with more than four fields:

```
NF > 4
```

Print every input line that does not have exactly four fields:

```
NF != 4
```

Print every input line in which the last field is greater than 4:

```
$NF > 4
```

Print the total number of fields in all input lines:

```
    { nf += NF }
END { print nf }
```

Print the total number of lines that contain `Beth`:

```
/Beth/ { nlines++ }
END    { print nlines }
```

Print the largest first field and the line that contains it (assumes some `$1` is positive):

```
$1 > max { max = $1; maxline = $0 }
END      { print max, maxline }
```

Print every line that has at least one field (that is, not empty or all spaces):

```
NF > 0
```

Print every line longer than 80 characters:

```
length($0) > 80
```

Print the number of fields in every line followed by the line itself:

```
{ print NF, $0 }
```

Print the first two fields, in opposite order, of every line:

```
{ print $2, $1 }
```

Interchange the first two fields of every line and then print the line:

```
{ temp = $1; $1 = $2; $2 = temp; print }
```

Print every line preceded by its line number:

```
{ print NR, $0 }
```

Print every line with the first field replaced by the line number:

```
{ $1 = NR; print }
```

Print every line after erasing the second field:

```
{ $2 = ""; print }
```

Print in reverse order the fields of every line:

```
{ for (i = NF; i > 0; i--) printf("%s ", $i)
  printf("\n")
}
```

Print the sums of the fields of every line:

```
{ sum = 0
  for (i = 1; i <= NF; i++) sum = sum + $i
  print sum
}
```

Add up all fields in all lines and print the sum:

```
    { for (i = 1; i <= NF; i++) sum = sum + $i }
END { print sum }
```

Print every line after replacing each field by its absolute value:

```
{ for (i = 1; i <= NF; i++) if ($i < 0) $i = -$i
  print
}
```